Digital Image Processing Laboratory: Model-Based Tomographic Reconstruction

January 24, 2019

1 Introduction

This laboratory explores the use of model based image processing for tomographic reconstruction. We will use MAP estimation and iterative coordinate descent techniques from lab one to reconstruct one tomographic slice from its associated sinogram data, which we assume has been acquired using parallel beam tomography.

As in previous MAP estimation problems, we will need three major components: the forward model, the prior model, and the optimization technique. The forward term in the MAP cost function models the scanner and is derived from the geometry of the system. As before, we will use a q-GGMRF prior model with an 8 point neighborhood system and circular boundary conditions with ICD optimization.

All solutions to this laboratory should be implemented in ANSI C. A skeletal code has been provided for you and you must complete a number of subroutines as specified by the section problems to complete the reconstruction algorithm.

2 System Geometry

This first section of the lab introduces the system geometry for parallel beam tomography. The object to be scanned is placed between the X-ray source and the detector array. The detector plane, denoted by the t-axis, is initially vertical and the view angle, θ, is 0 radians. As the source and detector rotate π radians about the object, the detector array collects information on the number of X-ray photon counts received. Given the Radon transform data, $p_\theta(t)$, as the output of a tomographic scan, we wish to reconstruct the optical density, $\mu(r_x, r_y)$, of the scanned object, where r_x and r_y are coordinates in Cartesian space.

In a continuous domain measurement system as in figure 1, the value of the Radon transform at point t_1 is the line integral of $\mu(r_x, r_y)$ along the ray parameterized by the point t_1 on the t-axis. The t-axis is discertized into a finite number of detectors indexed $d = 0, 1, \cdots, N_t - 1$ of width Δ_t as shown in figure 2. Projection data for each detector is collected for view angles $\theta_v \in \{v \times \Delta_\theta : v_i = 0, 1, \cdots, N_\theta - 1\}$, where v is the view index and $\Delta_\theta = \frac{\pi}{N_\theta}$ is the angle spacing. Therefore, the entire sinogram measurement vector, y, has $M = N_\theta \times N_t$ entries of which, the first N_t entries represent the detector responses collected in the first view and so forth. Due to this vectorization, the i-th element in y corresponds to detector index $d_i = mod(i, N_t)$ on view $v = floor\left(\frac{i}{N_t}\right)$. The center of the d-th detector

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

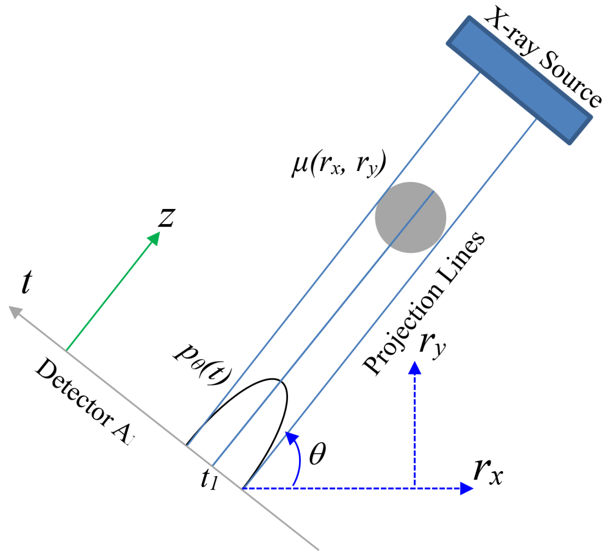

Figure 1: A geometric interpretation of how $p_\theta(t)$ relates to object $\mu(r_x, r_y)$. Projections for different t are calculated along the z axis. The (r_x, r_y) coordinate system is converted to the (z,t) system by the following transformation: $\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} \sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} z \\ t \end{bmatrix}$ where θ is the viewing angle.

is at $t_d = t_0 + d\Delta_t$. Detector d is the interval $\chi_d = \left[t_d^{start}, t_d^{stop}\right]$, where $t_d^{start} = t_d - \frac{\Delta_t}{2}$, and $t_d^{stop} = t_d + \frac{\Delta_t}{2}$. We assume that the origin of the t axis is at the center of the detector plane so the center of the detector with d index 0 is mapped to point $t_0 = -\frac{\Delta_t}{2}(N_t - 1)$.

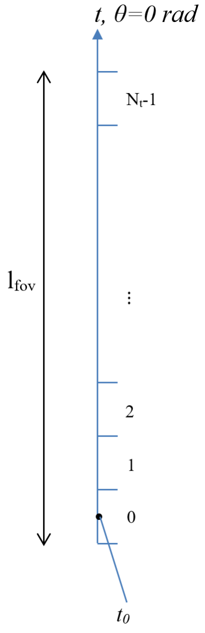

Figure 2: Discretization of the t-axis. Here the detector array is oriented as it would be in the initial view angle with $\theta = 0$ rad.

The field of view resulting from this geometry will have length $l_{fov} = N_t \times \Delta_t$, i.e., we will collect sinogram data for pixels in a square area of side l_{fov}. As the t-axis rotates about the object, its center traces out a circle which we will call the region of reconstruction. We will only reconstruct pixels that fall within the region of reconstruction. Other pixels in the field of view will be set to 0 or the attenuation coefficient of air.

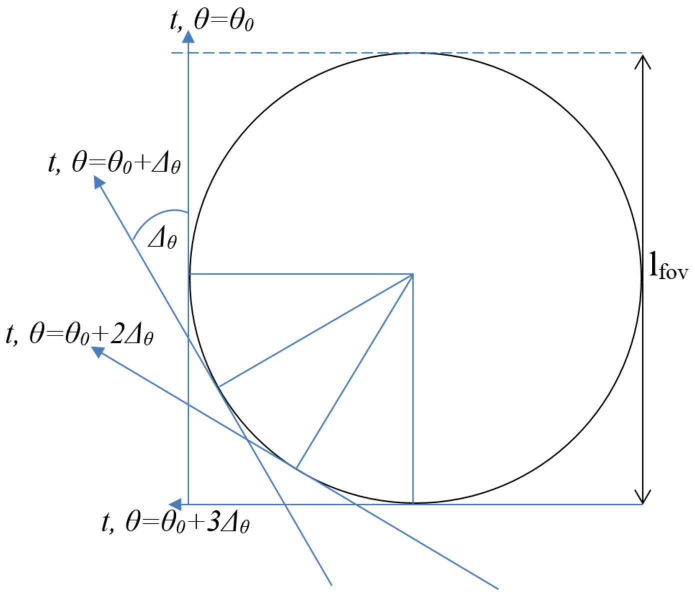

Figure 3: The field of view for the scanner.

The Cartesian space enclosed in the field of view has coordinate system (r_x, r_y). This space is divided into an N_x by N_y array of pixels, where $N_x = N_y$. The image area has a maximum width equal to l_{fov}, the field of view, which is also equal to the length of the detector array. Pixel j has center coordinates $(r_{j,x}, r_{j,y})$ where $j_x = mod(j, N_x)$, and $j_y = floor(j/N_x)$. Pixels are indexed from 0 to $N-1$, where $N = N_x \times N_y$ in raster order from left to right and bottom to top. $(r_{0,x}, r_{0,y})$ are the Cartesian coordinates of the center of the leftmost bottom pixel, where

$$r_{0,x} = -\frac{\Delta_p}{2}(N_x - 1), \qquad r_{0,y} = -\frac{\Delta_p}{2}(N_y - 1)$$

This is the first pixel in the image array. Each pixel is a square 2-cell in Cartesian space with width $\Delta_p = \frac{l_{fov}}{N_x}$. Pixel j has corresponding 2-cell denoted by $\Upsilon_j = \{(r_x, r_y) \in \mathbb{R} : |r_{j,x} - r_x| < \frac{\Delta_p}{2}, |r_{j,y} - r_y| < \frac{\Delta_p}{2}\}$.

The image vector x is a discrete representation of $\mu(r_x, r_y)$. Assuming that pixels are of uniform optical density, we can express $\mu(r_x, r_y)$ as

$$\mu(r) = \sum_{j=1}^{N} x_j u_j(r) \tag{1}$$

where r is an ordered pair, (r_x, r_y) in Cartesian space and $u_j(r)$ is a rect of width $\Delta_p \times \Delta_p$ centered around $(r_{j,x}, r_{j,y})$, i.e.,

$$u_j(r) = \begin{cases} 1 & \text{if } r \in \Upsilon_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

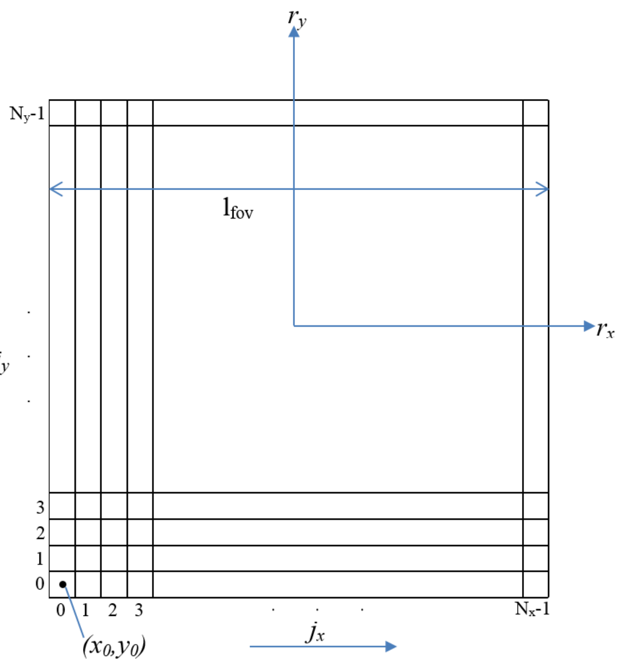

Figure 4: Discretization of Cartesian space into pixels. Pixel j is essentially a 2-D rect of width $\Delta_p \times \Delta_p$ and height equal to x_j, which is typically the average value of $\mu(r)$ over Υ_j.

Many of the system parameters discussed in this section are declared as members in structs. For example, struct type `SinoInfo` includes fields N_t, N_θ, etc. These structs are defined in data2d.h. In the code, we will pass pointers to the structs for compact notation. For a description of each of these fields, refer to the readme document in the forlder titled *C-code.*

3 MBIR Framework

Assume that the noisy sinogram data, Y is related to the noiseless image slice, X by

$$Y = AX + W \tag{3}$$

where pre-multiplication with matrix $A \in \mathbb{R}^{M\times N}$ represents the Radon transform of the optical density data modelled by X to sinogram data modelled by Y. $W \in \mathbb{R}^M$ is zero mean white additive Gaussian noise with diagonal covariance matrix R^{-1}. It can be shown that

$$\hat{x}_{map} = \arg\max_{x\in\Omega}\{p(x|y)\} = \arg\max_{x\in\Omega}\{\log p(y|x) + \log p(x)\} \tag{4}$$

where $p(y|x)$ is the probability of observing the measurement y, given the image x, and which encodes information about the sytem geometry and X-ray projection model. The prior model, $p(x)$, encodes the prior knowledge of how a tomographic image should behave.

Equation (3) presents a framework we are already familiar with from previous labs, the key difference being the manner in which the A-matrix is calculated based specifically on the system geometry of the tomographic scanner and this will be the topic of the next section.

4 Forward Model Formulation

4.1 Derivation of the Likelihood Function

Let $\Lambda \in \mathbb{R}^M$ be a random vector that models the photon measurement, where $\Lambda = \{\Lambda_i : i = 0, \cdots, M-1\}$. Λ_i models the number of photons received by detector d_i on view v_i in a noiseless environment as a Poisson random variable with expected value λ_i, i.e.,

$$\Lambda_i | x \sim \text{Poisson}(\lambda_i)$$

However, in the presence of electronic noise that we model as AWGN with variance σ_e^2, a more accurate distribution for Λ_i is given by

$$\Lambda_i \triangleq \text{Poisson}(\lambda_i) + \mathcal{N}(o, \sigma_e^2)$$

The sinogram measurement Y_i is given by

$$Y_i \triangleq -\log\left(\frac{\Lambda_i}{\lambda_{T,i}}\right) \tag{5}$$

where $\lambda_{T,i}$ is the photon count received during the air calibration scan. Therefore,

$$\mathbb{E}\left[\Lambda_i | x\right] = \text{Var}\left[\Lambda_i | x\right] = \lambda_i$$

Plugging this result into (5), we have that

$$\mathbb{E}\left[Y_i | x\right] = \mathbb{E}\left[-\log\left(\frac{\Lambda_i}{\lambda_{T,i}}\right)\right] \approx -\log\left(\mathbb{E}\left[\frac{\Lambda_i}{\lambda_{T,i}}\right]\right) = -\log\left(\frac{\lambda_i}{\lambda_{T,i}}\right) \tag{6}$$

Using the approximation that for an arbitrary random variable Z,

$$\text{Var}[f(Z)] \approx [f'(\mathbb{E}[Z])]^2 \text{Var}[Z] \tag{7}$$

we get

$$\text{Var}(Y_i | x) = \frac{\lambda_i + \sigma_e^2}{\lambda_i^2}$$

Furthermore, we assume that Y_i are Gaussian distributed and conditionally independent given x, i.e.,

$$Y | x \sim \mathcal{N}(\mu_{Y|x}, R^{-1})$$

where $R \in \mathbb{R}^{M \times M}$ is a diagonal matrix with

$$R_{i,i} = \frac{1}{\text{Var}[Y_i | x]}$$

Then it can be shown that

$$\log p(y|x) = -\frac{1}{2}(y - [Y|x])^T R (y - [Y|x]) + C \tag{8}$$

where C is a normalizing function that does not depend on x. The diagonal entries of R are stored in vector format on file and can be read into your program using the function *readweight* defined in *io2d.c*.

Section Problems:

1. Use $\mathbb{E}[\Lambda_i|x] = \lambda_i$, $\mathrm{Var}(\Lambda_i|x) = \lambda_i + \sigma_e^2$, and the approximation in eq (7) to show that $\mathrm{Var}(Y_i|x) = \frac{\lambda_i+\sigma_e^2}{\lambda_i^2}$.

4.2 Calculating the A-matrix Entries from Pixel Profiles

If detector d_i received a single X-ray beam incident on point τ on the t-axis during the v_i-th view, the number of photons received after attenuation through the object would have expected value

$$\mathbb{E}[\Lambda_i|\mu] = \lambda_{T,i} e^{-\int_{r\in\gamma_\tau} \mu(r)dr}$$

where γ_τ is the set of points, r, in Cartesian space that are traversed by the X-ray from the source to the detector. But detector d_i corresponds to an interval χ_{d_i} on the t axis so to calculate the total response of detector d_i, we must sum over all the X-rays incident upon it, and we get

$$\mathbb{E}[\Lambda_i|\mu] = \int_{\tau\in\chi_{d_i}} \lambda_{T,i} e^{-\int_{r\in\gamma_\tau} \mu(r)dr} d\tau$$

Then, from (6) we have that

$$\begin{aligned}\mathbb{E}[Y_i|\mu] &= -\log\left(\frac{\int_{\tau\in\chi_{d_i}} \lambda_{T,i} e^{-\int_{r\in\gamma_\tau} \mu(r)dr} d\tau}{\lambda_{T,i}}\right) = -\log\left(\int_{\tau\in\chi_{d_i}} e^{-\int_{r\in\gamma_\tau} \mu(r)dr} d\tau\right) \\ &\approx -\int_{\tau\in\chi_{d_i}} \log\left(e^{-\int_{r\in\gamma_\tau} \mu(r)dr}\right) d\tau = \int_{\tau\in\chi_{d_i}} \int_{r\in\gamma_\tau} \mu(r)drd\tau\end{aligned}$$

This result says that the i-th sinogram measurement is the sum of the line integrals of the attenuation coefficients along all the rays incident on detector d_i on view v_i.

Next, if we assume that pixels are of uniform density as we did in equations (1) and (2), we can replace $\mu(r)$ above with the expression in (1) and get

$$\begin{aligned}\mathbb{E}[Y_i|x] &= \int_{\tau\in\chi_{d_i}} \int_{r\in\gamma_\tau} \sum_{j=0}^{N-1} x_j u_j(r)drd\tau \\ &= \sum_{j=0}^{N-1} x_j \int_{\tau\in\chi_{d_i}} \int_{r\in\gamma_\tau} u_j(r)dr = \sum_{j=0}^{N-1} x_j \int_{\tau\in\chi_{d_i}} \int_{r\in\gamma_\tau\cup\Upsilon_j} dr \\ &= \sum_{j=0}^{N-1} A_{i,j} x_j\end{aligned}$$

Therefore,

$$A_{i,j} = \int_{\chi_{d_i}} \int_{ray, r\in\Upsilon_j} dr \qquad (9)$$

This result is shown pictorially in figure 5, and it says that $A_{i,j}$ is the sum of the lengths of intersections of all the rays that pass through pixel j and hit detector d_i.

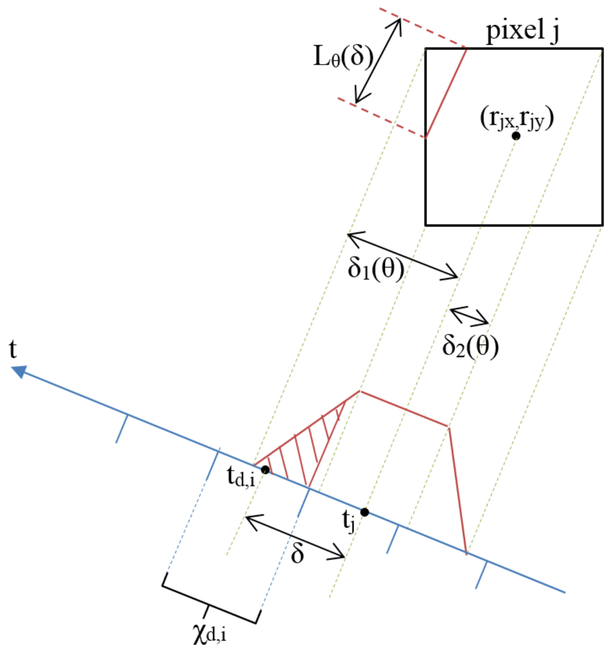

Figure 5: Generation of the profile of pixel j at view θ. Note that the projection has non-zero values for a small and contiguous portion of the detector array. The shaded area where the pixel profile of j and detector interval χ_{d_i} overlap is the value of $A_{i,j}$

4.2.1 Calculating the Pixel Profiles

The projection of a pixel of uniform density, as shown in figure (5), will have a trapezoidal profile, except for at viewing angles of 0 and $\frac{\pi}{2}$ rad, where the profile will be square, and at angle $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ rad, where the profile is triangular. We will refer to these projections as pixel profiles. In order to calculate $A_{i,j}$ entries we first need to complete an intermediate step, which is to compute all the possible pixel profiles.

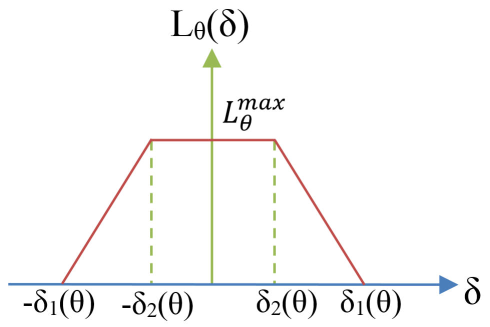

Figure 6: Pixel profile, L_θ, produced at view angle θ plotted as a function of δ.

Let us define some points of interest on the t-axis. Let the X-ray passing through the center of pixel j be incident on the t-axis at t_j. Let t_{d_i} be the t-axis coordinate of the center of detector d_i. Let δ be the distance between t_j and t_{d_j}. Then the line integral of the intersection between the X-ray and pixel j, namely L_θ, may be plotted as a function of δ, where $L_\theta(\delta)$ is a trapezoid parameterized by $\delta_1(\theta)$, $\delta_2(\theta)$, and L_θ^{max}.

In parallel beam tomography, all pixel profiles for a given θ will have the same trapezoidal parameters but will be centered at different points on the t-axis. So it will suffice to calculate one pixel profile per view, which can later be translated to the appropriate spot on the t-axis, given pixel index j. The paramters can be calculated for different θ as shown in table 1.

θ	$L_\theta^{\max}$	$\delta_2(\theta)$	$\delta_1(\theta)$
$\left[0, \frac{\pi}{4}\right)$	$\frac{\Delta_p}{\cos\theta}$	$\frac{\Delta_p}{\sqrt{2}}\left\lvert\sin\left(\frac{\pi}{4}-\theta\right)\right\rvert$	$\frac{\Delta_p}{\sqrt{2}}\cos\left(\frac{\pi}{4}-\theta\right)$
$\left[\frac{\pi}{4}, \frac{\pi}{2}\right)$	$\frac{\Delta_p}{\sin\theta}$		
$\left[\frac{\pi}{2}, \frac{3\pi}{4}\right)$	$\frac{\Delta_p}{\sin\theta}$	$\frac{\Delta_p}{\sqrt{2}}\left\lvert\cos\left(\frac{\pi}{4}-\theta\right)\right\rvert$	$\frac{\Delta_p}{\sqrt{2}}\sin\left(\theta-\frac{\pi}{4}\right)$
$\left[\frac{3\pi}{4}, \pi\right)$	$\frac{\Delta_p}{-\cos\theta}$		

Table 1: Pixel profile parameters as a function of the view angle θ.

Assuming the detector is uniformly sensitive, $A_{i,j}$ is the area under the pixel profile of the j-th pixel where it overlaps with the d_i-th detector, i.e.,

$$A_{i,j} = \int_{t_{d_i}^{start}}^{t_{d_i}^{stop}} L_\theta(t-t_j)dt = \int_{t_{d_i}^{start}-t_j}^{t_{d_i}^{stop}-t_j} L_\theta(\delta)d\delta \tag{10}$$

Pixel profiles are widest at $\theta = \frac{\pi}{4}$ rad and $\theta = \frac{3\pi}{4}$ rad when the profiles are triangular. The maximum width is given by $2\delta_1^{\max} = \Delta_p\sqrt{2}$ and this width will span at most $d^{\max} = ceil\left(\frac{2\delta_1^{\max}}{\Delta_t}\right) + 1$ detectors. All the other detectors will have a response of zero from this pixel at this view. Figure (7) shows how most of the elements in a column of A are zeros. This makes A a sparse matrix overall.

To compute and store the pixel profiles, we will take an interval on the t-axis centered about $t = t_j$ or equivalently, $\delta = 0$ in figure (6). In order to capture the entire pixel profile, we will choose our interval to be two pixels wide since

$$2\delta_1 \leq \Delta_p\sqrt{2} < 2\Delta_p$$

and discretize it into `LEN_PIX` $= 511$ elements. For the starting point of each of these discrete elements, we will calculate a value for $L_{\theta_{v_i}}(\delta_k)$, $\delta_k = k = 0, \cdots, 510$, and this will be done for each view, v_i. The result will be stored in a 2D block of memory with pointer `**profiles`, where row v_i will contain values of $L_{\theta_{v_i}}$ at `LEN_PIX` points, i.e.,

$$profiles[v_i][k] = L_{\theta_{v_i}}(\delta_k), \qquad \delta_k = \frac{2\Delta_p k}{LEN\_PIX} - \Delta_p,$$
$$k = 0, 1, \cdots, LEN\_PIX - 1$$

where Δ_p is subtracted from $\frac{k}{LEN\_PIX}$ to translate the center of a $2\Delta_p$ wide interval to $\delta = 0$.

An N_θ by `LEN_PIX` block of memory has already been allocated for `**profiles` in the skeleton code. The syntax for defining the *pix_prof_comp* routine is:

```
void  pix_prof_comp(
  struct SinoInfo *sino_info,  /* pointer to scanner parameters  */
  struct ImgInfo *img_info,    /* pointer to image parameters  */
  float **pix_prof             /* pointer to 2D array of allocated memory
                                    with N_theta by LEN_PIX elements */
  )
```

where pointers `*psino_info` and `*pimg_info` can be passed to the function from main and should already contain the parameters needed to calculate the pixel profiles. `LEN_PIX` is defined as a macro in *data2d.h*. After inserting the function call in the appropriate place in the skeletal code, running the code should write the profiles in raster order to $shepp\backslash output\backslash shepp\_mbir.prf$, which can be viewed using the matlab code $display\_profiles.m$ in the *matlab* folder.

Section Problems:

1. Write a function for `pix_prof_comp`.
2. Call your function in the appropriate place in the skeleton code.
3. The pixel profiles will be written to file as *mbir_shepp.prf* in *./recon* and can be viewed using the matlab code *display_profiles.m*. Include the matlab figure of the resulting mesh plot of the pixel profiles in your report.

4.2.2 Computing and Accessing $A_{*,j}$

The *data2d.h* file declares a struct Acol to store cloumns of A-matrix with three fields: val is a pointer to an array of the non zero elements of $A_{*,j}$, index points to an array that stores i indices of the corresponding values in val, and n_index keeps track of the total number of non zero elements in $A_{*,j}$.

In order to store the entire A-matrix in memory, a 2-dimensional pointer, `**A_col_arr` has been defined in the skeletal code in the size of the image. Each of these adresses will point to a struct of type `Acol` as in figure (8).

You are to calculate the A-matrix one column at a time. To do this, you must write a function `A_col_comp`. The syntax for the routine should be:

```
void  A_col_comp(
  int im_row
  int im_col
  struct SinoInfo *sino_info,  /* pointer to scanner parameters  */
  struct ImgInfo *img_info,    /* pointer to image parameters  */
  float **pix_prof,             /* pointer to 2D array of allocated memory
                                   with N_theta by LEN_PIX elements */
  struct Acol *A_col
  )
```

where `im_row` and `im_col` are the row and column numbers of the pixel with index j and we wish to compute $A_{*,j}$. `*A_col` points to an array that has been preallocated with d_{max} addresses for its `*val` and `*index` fields. This array is to act as a placeholder and will be allocated $d^{\max} \times N_\theta$ addresses to accomodate the maximum possible number of non zero $A_{*,j}$ entries. The function is called for every pixel in the image and values from `A_col` are copied to the structs in the addresses of `A_col_arr`.

There will still be unused memory at the end of `A_col->val` because `A_col->n_index` $< d^{\max} \times M$ and we are using some additional memory to store indices of non zero elements but the memory allocated in this manner is still much less than you would have needed to store all M entries of $A_{*,j}$, most of which would be 0 and skipping these zeros will save on computation as well.

Now, assume that the detector sensitivty is uniform across its surface so if we distretize each detector into `LEN_DET` = 101 elements, each one is equally sensitve so we assign each a sensitivity weight of 1/`LEN_DET` and represent it as a vector ϕ of sensitivities. The t-axis coordinates of the starting points of each of these elements may be indexed as

$$t_h = t_0 - \frac{\Delta_t}{2} + d_i\Delta_t + \frac{h\Delta_t}{LEN\_DET}, \qquad h = 0, \cdots, LEN\_DET - 1 \tag{11}$$

where $t_0 - \frac{\Delta_t}{2}$ is the start of detector 0, d_i is the detector index, and the addition of $\frac{h\Delta_t}{LEN\_DET}$ is to translate the point to the start of each detector element.

At view angle θ_v, the corresponding pixel profile index may be calculated as

$$l_h = ceil\left([t_h - (r_{jy}\cos\theta_v - r_{jx}\sin\theta_v) + \Delta_p]\frac{LEN\_PIX}{2\Delta_p}\right) \tag{12}$$

where (r_{jx}, r_{jy}) are the Cartesian coordinates of the center of pixel j. $(r_{jy}\cos\theta_v - r_{jx}\sin\theta_v)$ is subtracted to find δ, Δ_p is added to translate the $\delta = 0$ point to the center of the $2\Delta_p$ interval over which the pixel profile was calculated, and δ is sclaed by $\frac{LEN\_PIX}{2\Delta_p}$ and rounded up to find the corresponding k index. Values for k should be only be considered if they are between 0 and $LEN\_PIX - 1$.

Then $A_{i,j}$ is the following sum:

$$\begin{aligned} A_{i,j} &= \sum_{h=0}^{LEN\_DET-1} \phi(h) \times profiles[v_i][l_h] \\ &= \frac{1}{LEN\_DET} \sum_{h=0}^{LEN\_DET-1} profiles[v_i][l_h] \end{aligned}$$

If the value is non zero, it can be stored in an adress of `A_col->val`, its index, i can be stored in an address in `A_col->index`, and the total number of non zero elements can be stored in `A_col->n_index`. The information in this section is summarized in the following pseudocode.

Algorithm to Compute A-matrix column $A_{*,j}$

```
1: $(r_x, r_y) \leftarrow$ Cartesian coordinates of center of pixel $j$;
2: $A\_col\text{->}n\_index \leftarrow 0$;
3: for every view $v = 0, 1, \cdots, N_\theta - 1$ do
4:     /* $i$ index for 0th detector in $v$th view */
5:     $ind \leftarrow v \times N_t$;
6:     /* find $t$ axis points corresponding to $\delta_{\min}$ and $\delta_{\max}$ */
7:     $t_{\min} \leftarrow r_y \cos\theta_v - r_x \sin\theta_v - \Delta_p$;
8:     $t_{\max} \leftarrow r_y \cos\theta_v - r_x \sin\theta_v + \Delta_p$;
9:     /* find corresponding detectos indices */
10:    $d_{\min} \leftarrow max\{d_{t_{\min}}, 0\}$;
11:    $d_{\max} \leftarrow min\{d_{t_{\max}}, N_t\}$;
12:    for every detector $d = d_{\min}, d_{\min} + 1, \cdots, d_{\max}$ do
13:        $i \leftarrow ind + d$;
14:        $Aval \leftarrow 0$ ;
15:        for every detector element $h = 0, 1, \cdots, LEN\_DET - 1$ do
16:            $t_h \leftarrow t_0 - \frac{\Delta_t}{2} + d\Delta_t + \frac{h\Delta_t}{LEN\_DET}, \qquad h = 0, \cdots, LEN\_DET - 1$;
17:            $l_h \leftarrow ceil\left([t_h - (r_y \cos\theta_v - r_x \sin\theta_v) + \Delta_p]\frac{LEN\_PIX}{2\Delta_p}\right)$;
18:            if $0 \leq l_h \leq LEN\_PIX$ then
19:                $Aval+ = \frac{profiles[v][l_h]}{LEN\_DET}$;
20:            end if
21:        end for
22:        if $Aval \geq 0$ then
23:            $A\_col\text{->}val[A\_col\text{->}n\_index] \leftarrow Aval$;
24:            $A\_col\text{->}index[A\_col\text{->}n\_index] \leftarrow i$;
25:            $A\_col\text{->}n\_index + +$ ;
26:        end if
27:    end for
28: end for
```

Section Problems:

1. Write a function for `A_col_comp`.
2. Call your function in the appropriate place in the skeleton code.
3. Compute the initial error sinogram, $e = y - Ax$, where x is set to the initial conditions.
4. The error sinogram term will be written to file as *mbir_shepp.err* in *./recon* and can be viewed using the matlab code *display_err.m*. Include the matlab figure of the resulting mesh plot of the initial error term in your report.

5 Prior Model

The image x will be modeled using a qGGMRF with circular boundary conditions and parameter g_s as in the following array

0.11	0.14	0.11
0.14	0	0.14
0.11	0.14	0.11

and probablity density function

$$p(x) = \frac{1}{z}\exp\left\{-\sum_{\{s,r\}\in\mathcal{C}} b_{s,r}\rho(x_s - x_r)\right\}$$

where $g_{s,r} = g_{|s-r|} = b_{s,r}$, z is a normalizing constant independent of x, and

$$\rho(x_s - x_r) = \frac{|x_s - x_r|^p}{p\sigma_x^p}\left(\frac{\left|\frac{x_s-x_r}{\sigma_x T}\right|^{q-p}}{1+\left|\frac{x_s-x_r}{\sigma_x T}\right|^{q-p}}\right)$$

Therefore, we can rerwrite the log prior term as

$$-\log p(x) = \sum_{\{s,r\}\in\mathcal{C}} b_{s,r}\rho(x_s - x_r) + \log z \tag{13}$$

6 Cost Function Formulation and ICD Optimization

We can now formulate the MAP cost function using the log likehood term in (8) and the log prior term from (13) into equation (4). Dropping terms independent of x, we obtain the following cost function to minimize,

$$c(x) = \frac{1}{2\sigma_W^2}||y - Ax||_R^2 + \frac{1}{p\sigma_x^p}\sum_{\{s,r\}\in\mathcal{C}} b_{s,r}|x_i - x_j|^p\left(\frac{\left|\frac{x_s-x_r}{T\sigma_x}\right|^{q-p}}{1+\left|\frac{x_s-x_r}{T\sigma_x}\right|^{q-p}}\right) \tag{14}$$

Using the majorization techniques of lab 1, the MAP surrogate cost function becomes

$$Q(x;x') = \frac{1}{2\sigma_W^2}||y - Ax||_R^2 + \sum_{\{s,r\}\in\mathcal{C}} \tilde{b}_{s,r}(x_s - x_r)^2 \tag{15}$$

where

$$\tilde{b}_{s,r} \leftarrow \begin{cases} \frac{b_{s,r}}{p\sigma_x^q T^{q-p}} & \text{if } x'_s - x'_r = 0 \\ b_{s,r}\frac{|x'_s-x'_r|^{p-2}}{2\sigma_x^p}\frac{\left|\frac{x'_s-x'_r}{T\sigma_x}\right|^{q-p}\left(\frac{q}{p}+\left|\frac{x'_s-x'_r}{T\sigma_x}\right|^{q-p}\right)}{\left(1+\left|\frac{x'_s-x'_r}{T\sigma_x}\right|^{q-p}\right)^2} & \text{else} \end{cases}$$

The ICD algorithm to optimize the surrogate cost function is as follows

ICD Algorithm with Majorization of GGMRF Prior:

```
1: Initialize $x \leftarrow$ initial condition;
2: Calculate pixel profiles;
3: Calculate forward projection matrix, $A$;
4: initialize $e \leftarrow y - Ax$;
5: set max number of iterations, $K$;
6: for $k = 0, 1, \cdots, K-1$ and while average update > stop threshold do
7:     randomize the order of pixels $j \in \mathcal{S}$ for fast convergence;
8:     total update $\leftarrow 0$;
9:     for $j = 0, 1, \cdots, N-1$ do
10:        if (($j \in$ region of reconstruction) && ($A_{*,j}$ is non empty)) then
11:            $\tilde{b}_{j,k} \leftarrow b_{j,k} \frac{\rho'(x'_j - x'_k)}{2(x'_j - x'_k)} \quad \forall k \in \delta j$;
12:            $\theta_1 \leftarrow -e^T R A_{*,j} + \sum_{k \in \delta j} 2\tilde{b}_{j,k}(x_j - x_k)$;
13:            $\theta_2 \leftarrow ||A_{*,j}||_R^2 + \sum_{k \in \delta j} 2\tilde{b}_{j,k}$;
14:            $\alpha^* \leftarrow clip\left\{-\frac{\theta_1}{\theta_2}, [-x_j, \infty)\right\}$;
15:            $x_j \leftarrow x_j + \alpha^*$;
16:            total update = total update + $|\alpha^*|$;
17:            $e \leftarrow e - A_{*,j}\alpha^*$;
18:        end if
19:    end for
20:    average update $\leftarrow$ total update$/N$;
21: end for
```

Section Problems:

1. Produce a reconstruction of the tomographic slice by performing upto 10 ICD iterations or until the average update is smaller than the stopping threshold.

2. The image will be written to file as *shepp_mbir.rec* in *shepp/output*. Use the Matlab script *display_image.m* to view it and include the figure in your report.

3. Plot the cost function of (14) as a function of the iteration number for the reconstruction of $\hat{x}_{MAP}$.

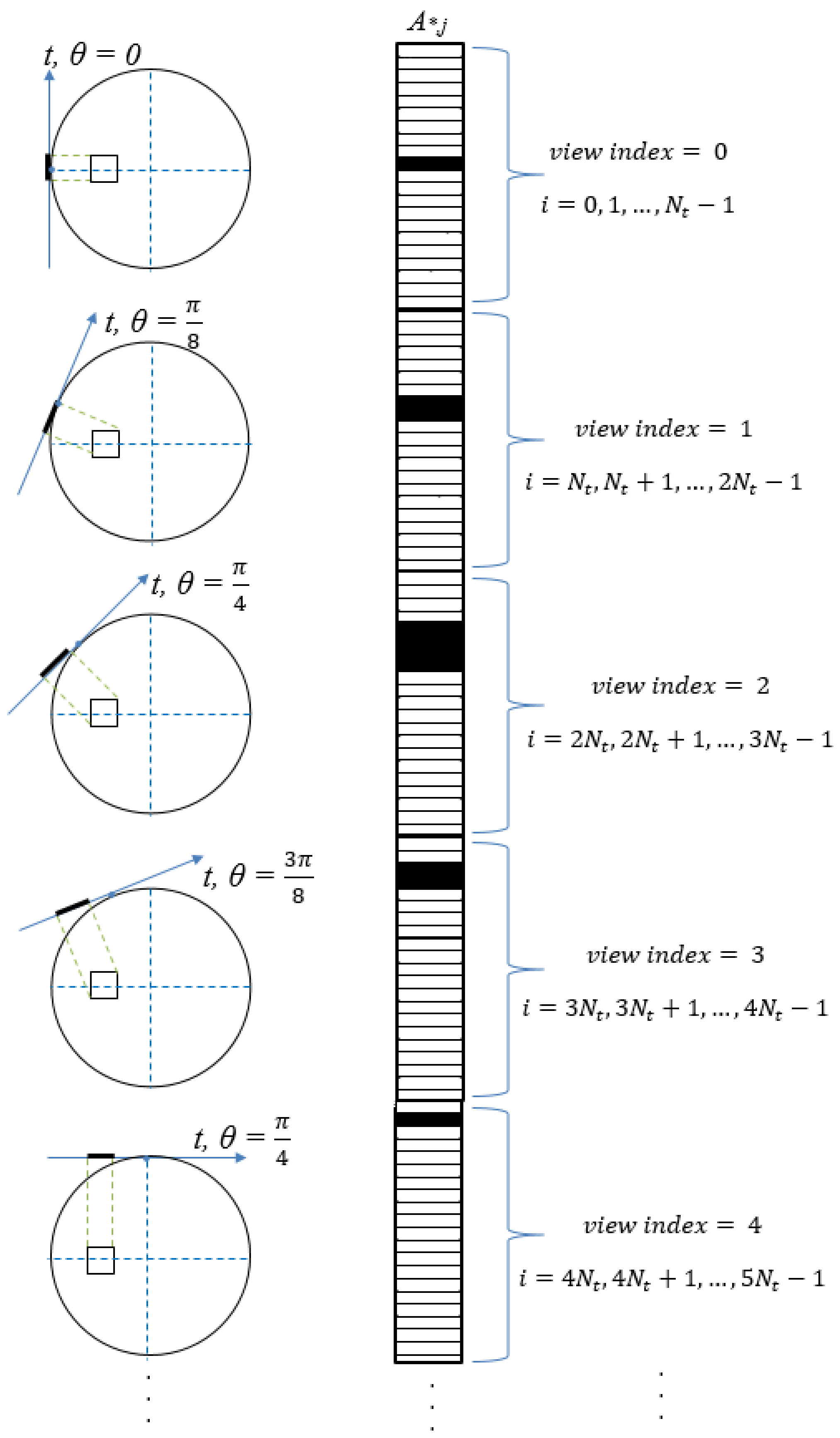
A_{*j}
$t,\ \theta = 0$
$t,\ \theta = \frac{\pi}{8}$
$t,\ \theta = \frac{\pi}{4}$
$t,\ \theta = \frac{3\pi}{8}$
$t,\ \theta = \frac{\pi}{4}$
view index = 0
$i = 0, 1, \dots, N_t - 1$
view index = 1
$i = N_t, N_t + 1, \dots, 2N_t - 1$
view index = 2
$i = 2N_t, 2N_t + 1, \dots, 3N_t - 1$
view index = 3
$i = 3N_t, 3N_t + 1, \dots, 4N_t - 1$
view index = 4
$i = 4N_t, 4N_t + 1, \dots, 5N_t - 1$

Figure 7: Column $A_{*,j}$ for view indices 0 through 5 assuming $\Delta_\theta = \frac{\pi}{8}$. The empty cells represent $A_{i,j}$ elements that are zero and the black cells are non zero entries. t_j, the position of the center of the profile on the t axis, varies sinusoidally with θ, hence the term sinogram.

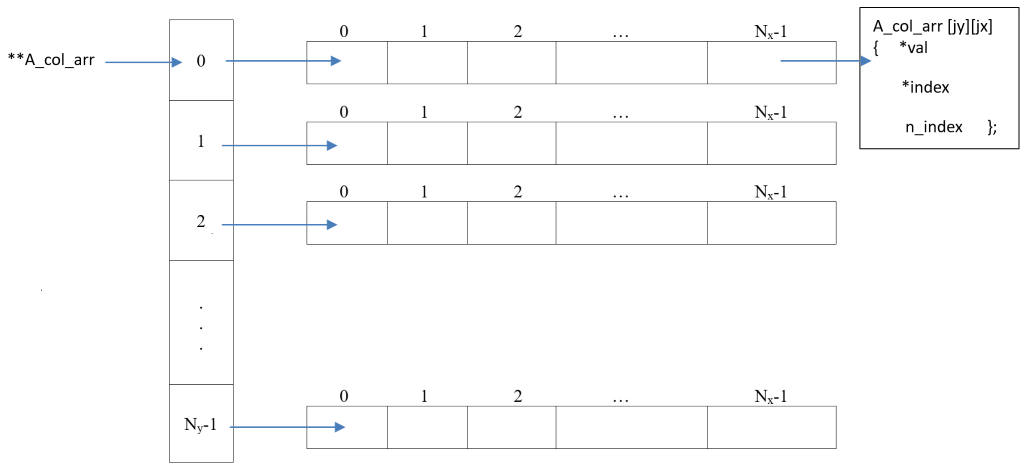

Figure 8: Memory allocation for the A matrix. Each address points to a struct of type `Acol`.